PhUSE 2017

Paper TT01

Helping The Define.xml User

Dave Iberson-Hurst, Assero Limited, Teignmouth, United Kingdom

ABSTRACT

The FDA often comment at industry gatherings on the quality of define.xml files received as part of submissions; either they are of low quality or are missing! This presentation will look at tooling designed to help users build better define.xml files.

When creating a define.xml file, there can be a lack of source information: just an annotated CRF, possibly an existing define file from a previous study or there might be a full electronic study design. The presentation will discuss in detail:

- How graph technologies can help;
- How using a semantic metadata repository can aid the process in particular with Value Level Metadata;
- How the tool can "learn" as define files are created to aid future work and guide and assist a user;
- Allow for easy upgrades from one version of define.xml to another;

All focused around easing the user's task and creating higher-quality define.xml files.

INTRODUCTION

At the PhUSE Computational Science Symposium in March 2017, held in Silver Spring, the FDA presented a slide, see ***Figure 1***, which contained some interesting statistics.

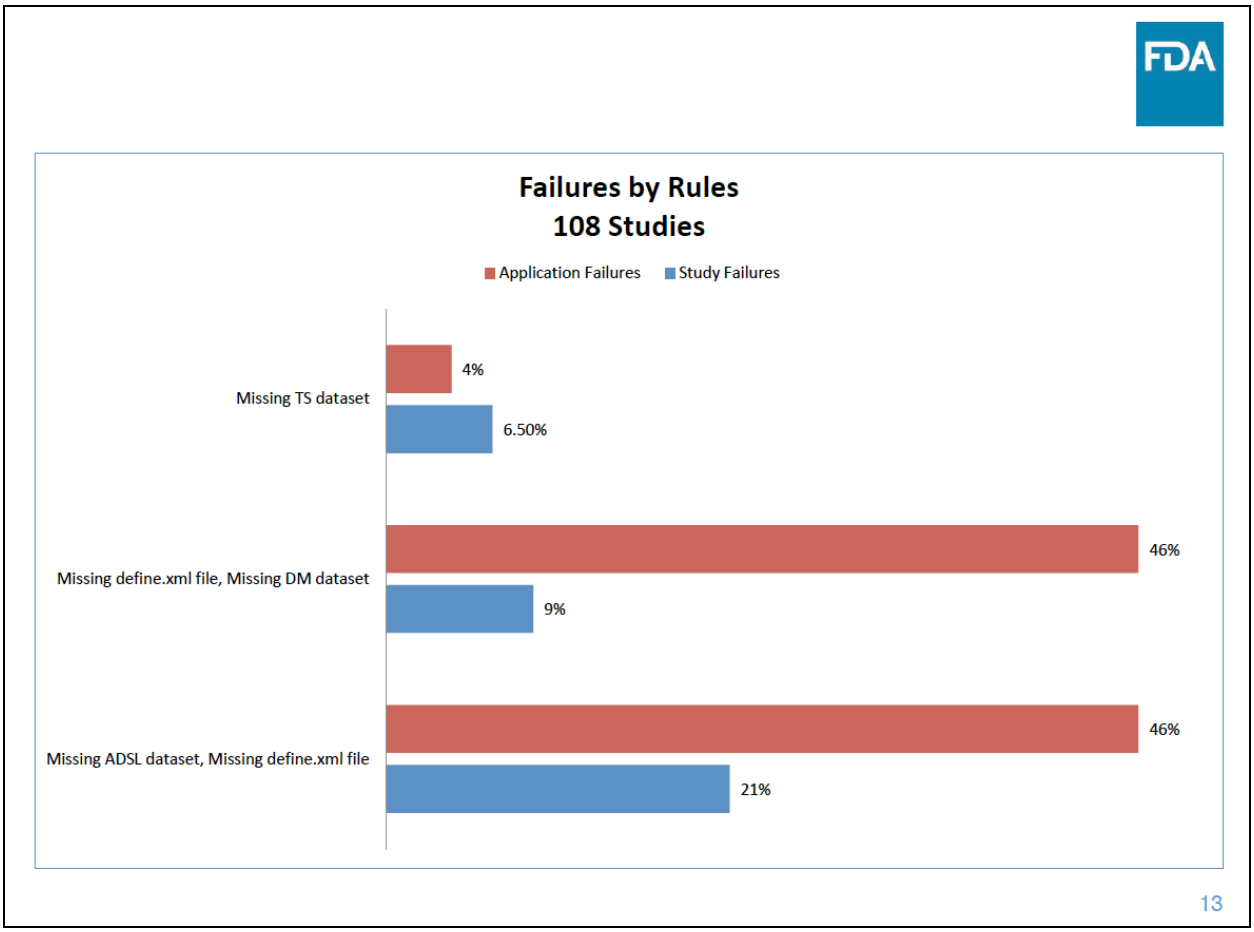

Figure 1: Slide from FDA Presentation at the 2017 CSS Meeting

The slides were presented by Crystal Allard, Special Assistant to the Director Office of Computational Science (Allard, 2017) and related to a new eCTD rule requiring a Demographics Dataset and Define.xml be submitted in module 4. The FDA examined 108 studies that they already had in Clinical Data Interchange Standards Consortium (CDISC) Study Data Tabulation Model (SDTM) format and checked conformance against the new rule. 9% of studies failed the check, while 46% of applications to which those studies belonged failed.

Now, on first examination, that statistic is rather damning. As with all presentations, it should be remembered that context is everything. This slide is referenced simply as an illustration of the continuing issues being encountered within industry, when using the define.xml standard (CDISC, 2005-2017). The author remembers issues encountered early in the adoption of define.xml with schema validation and contributing to the CDISC paper on the topic (CDISC, 2009). We don't appear, as an industry, to have got to grips with using the standard.

It is against this background that work commenced assisting users to build better define.xml files for use in submissions. It should be noted that this paper focuses on the use of a define.xml file to describe a SDTM submission and datasets. It does not cover the Analysis Data Model (ADaM) use case. The ADaM use case will form the basis for a second phase of the work.

AIMS

The purpose of the work is to create a tool that allows users to create better define.xml files with less effort. To this end there are several drivers to the work:

- Elevate the user above the XML and work at a more logical level
- Focus on what is required from a content perspective
- Present information in a manner that an average user understands
- Allow for different levels of source material
- Allow all versions of the standard to be supported

The aim of the tool is that the user should never see the XML; it simply should not be necessary. The define.xml file format is confusing for those that are not familiar with XML and the technologies involved. We need to promote the user to the logical level and present entities that they are familiar with such as Domains and Variables. Such information should be presented in a manner that they are familiar with as subject matter experts.

One of the issues with creating define.xml files is the starting point. It is recognized that there are different starting points and variations in the source material available to a user when a define.xml is created. There may be a full study design available, in an electronic form, from which information can be extracted. Alternatively there maybe an existing define.xml file. Some organizations use a super-set define.xml that is then 'constrained' to the study in question. Another approach is, of course, take the define.xml from the last study and amend, the cut-and-paste approach. Finally, the worse case scenario, where we have little source information except a paper annotated Case Report Form (aCRF).

It should also be recognized that the define.xml may also be being created for a variety of different use cases at different points in the life-cycle and not just for use within a submission context. One popular use case is using a define.xml file as a data specification for datasets that are to be returned by a Contract Research Organization (CRO)

LOGICAL VIEW

With the aim of presenting a more user-friendly view to the average user, a logical view of the content that is placed into a define.xml file was developed.

A define file is, in essence, a description of a study. We describe the study in terms of the domains used, the variables, and associated terminologies along with clarifying information such as how some variables are derived. It allows you to understand the construction of a study without having to look at the data.

The define.xml file is no different to the label you will find on a bottle of wine. The label tells you what is inside the bottle, the country it was produced in, the vineyard, the grape(s) used and the year in which it was produced. That label provides you with enough information about the wine to allow you to make an informed decision about whether to buy it or open it. You don't have to open the wine, though we might like to try the contents. Define.xml provides the same function for the study data describing the data, without the need to crawl all over it.

Figure 2 presents the logical view developed. Walking through the diagram from the top left, we see the study that has several properties that include the simple Name, Description and Protocol Description.

As an aside, of course we see the notion studies in other models and contexts; say a study design, an ODM containing data captured from a study. So we see a need for a common model 'fragment' for a study that would contain all the possible properties for a study that could then be incorporated into other models such that we get consistency across models and applications.

Returning to the logical view we see associated with the study is the aCRF and maybe one or more supplemental documents such as the Reviewer's Guide. The study contains one or more Domains with each Domain again having the expected properties such as Prefix (Code), Class, Purpose etc. Domains might be related. Either they may have supplemental domains or be split.

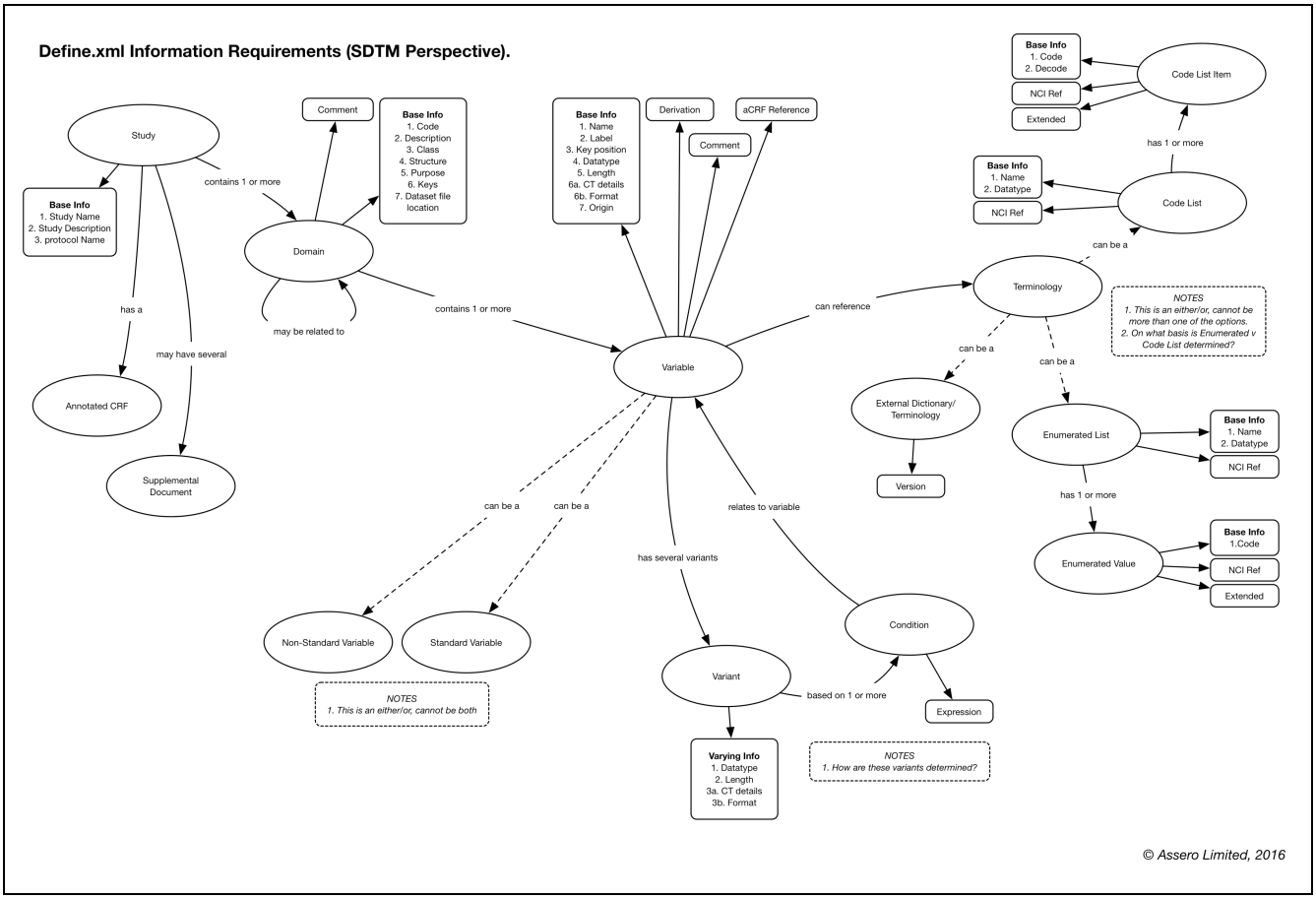

Figure 2: SDTM Define.xml Logical Model

Moving on from the domain, each contains one or more variables with each having a set of properties such as Name, Label, Datatype etc. In addition to the basic parameters, a variable can have a Comment and a Derivation along with an associated aCRF page reference. A variable maybe a standard or a non-standard variable.

As we are all aware a variable may be associated with some terminology. This could either be CDISC defined, sponsor defined, or a reference to an external terminology such as MedDRA. As from Define version 2.0.0, we also have the ability to refer to terminology terms as code/decode pairs or as an enumerated list.

While we can always force information into rectangular structures and define relationships between those rectangular structures as in relational databases, it may not be the best approach. We can already see the non-rectangular nature of the information emerging from the model. There is one piece that is yet to be covered that of the variation in a variable's properties for a set of conditions and what has generally been referred to as Value Level Metadata (VLM). We will examine this particular feature in detail to further the case that graph technology can help us in better tooling for define.xml.

So as to illustrate the point we can look at examples of some simple variations that we would encounter in a define file, look at the XML and then step back a little to look at the structure of the information content that define.xml demands of us.

Examples of some simple Vital Signs domain VLM are:

- VSORRES
  - datatype=”float”, length=”4”, significant digits=”1” when VSTESTCD = HEIGHT
  - datatype=”float”, length=”5”, significant digits=”1” when VSTESTCD = WEIGHT
- VSORRESU
  - datatype=”text”, length=”5”, code list=”IN” or “cm” when VSTESTCD = HEIGHT
  - datatype=”text”, length=”4, code list=”LB” or “kg” when VSTESTCD = WEIGHT

In Laboratory observations the conditions can become more complex with several conditions.

- LBORRES
  - datatype=”float”, length=”4”, significant digits=”1” when LBTESTCD=”GLUC” and LBCAT=”URINALYSIS” and LBSPEC=”BLOOD”
  - datatype=”text”, length=”8”, when LBTESTCD=”GLUC” and LBCAT=”URINALYSIS” and LBSPEC=”URINE” and LBMETHOD=”DIPSTICK”
  - datatype=”text”, length=”8” when LBTESTCD=”GLUC” and LBCAT=”URINALYSIS” and LBSPEC=”URINE” and LBMETHOD=”QUANT”

Figure 3 is very crowded but details the structure of the above VLM for the VSORRES and VSORRESU, as we would see it in the XML of a define.xml file. Each rectangle represents an XML element with the solid arrows representing element parent-child relationships. Dotted arrows represent define.xml “reference-definition” relationships, where an element uses an ODM Identifier (OID) to reference another element. Dotted rectangles represent element values.

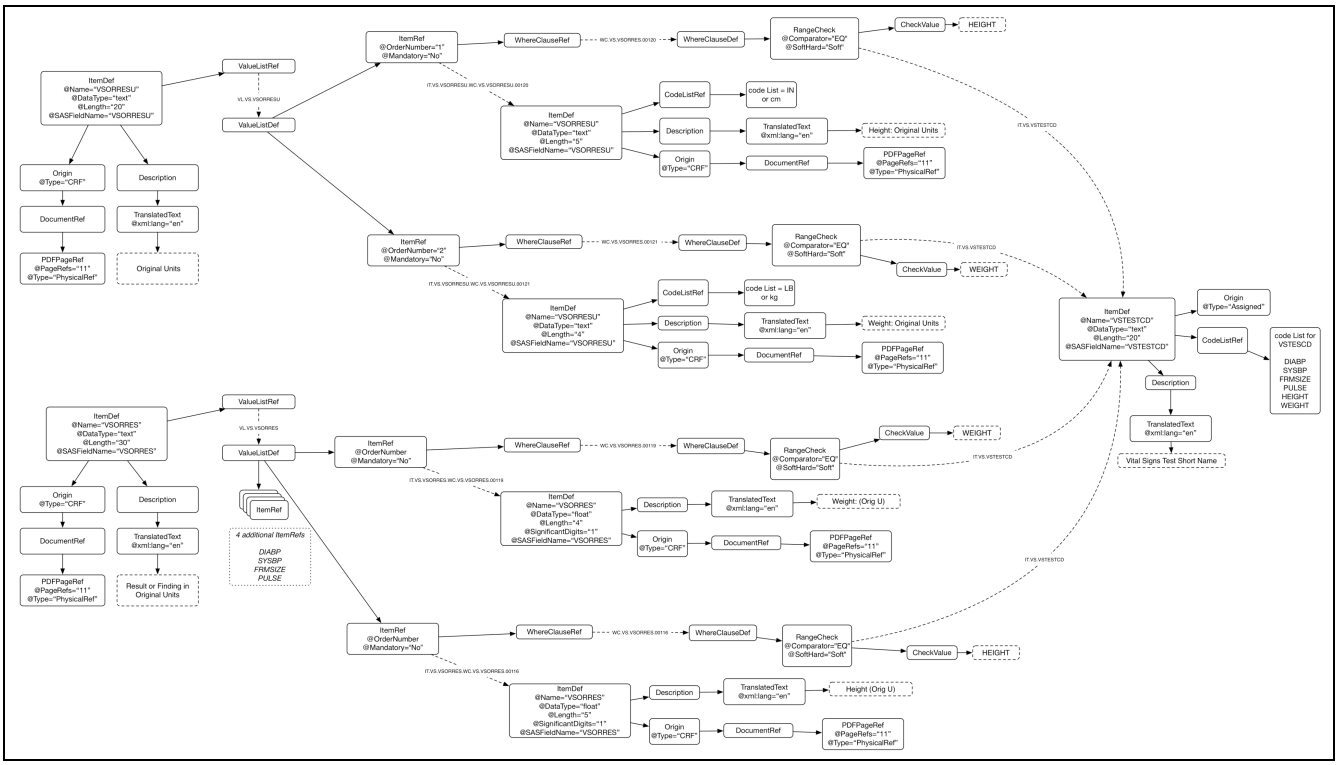

Figure 3: Value Level Metadata in Define.xml

At the left of the figure we see two sets of elements in the same form, the top one is shown in ***Figure 4*** so that it can be seen more clearly. It is the definition of the VSORRESU variable and it contains the generic property definitions for

the variable. The block below it on the left hand side in Figure 3 follows the same pattern and is the definition for VSORRES.

Looking at ***Figure 4*** we see the link from the ItemDef to a ValueListDef indicating that this variable has Value Level Metadata and we see the two arrows indicating two variants for this variable. Refer back to the bullet lists for VSORRESU above to see these variants.

One of the variants for VSORRESU is then shown in ***Figure 5***. As we can see there are two parts: a) the upper WhereClauseRef, WhereClauseDef and athe ssociated child elements and b) the bottom ItemDef and the chain of child elements stretching to the right that define the properties for the variant. The elements at the top define the conditions for which the variant is valid; in this case the variant is VSORRESU as a text variable, length 5 with code list for inches and centimeters is valid when “something” has a value of HEIGHT. That “something” is referenced by the dotted line and is the ItemDef for VSTESTCD. The elements for VSTESTCD are shown in ***Figure 6***. Note that the pattern we see in ***Figure 6*** is exactly the same as for VSORRESU shown in ***Figure 4*** but has an additional feature in that it references a code list of all the applicable Vital Sign Test Codes in use.

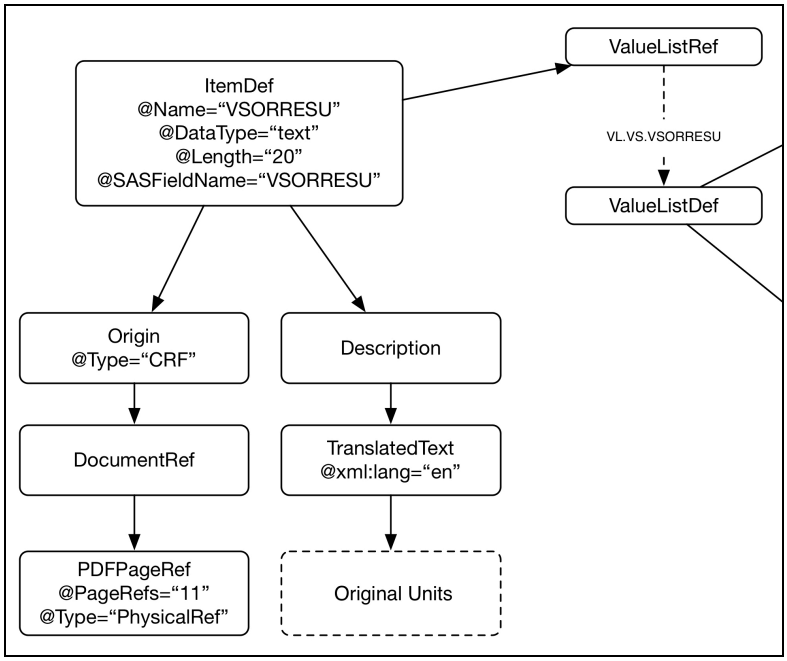

Figure 4: VSORRESU in Define.xml

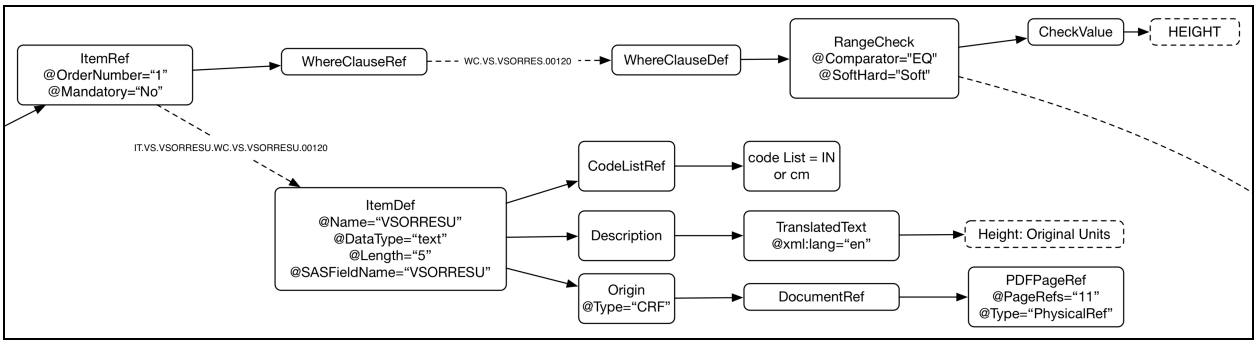

Figure 5: VSORRESU Variant in Define.xml

*Note: In **Figure 5** you well see a CodeListRef and a link to a statement about the contents of the code list so referenced. No attempt has been made to detail all of the elements that form the code list itself. For the purposes of this discussion it is enough to have knowledge of the code list content. The XML for a code list is sizable and would have made the figure too complex.*

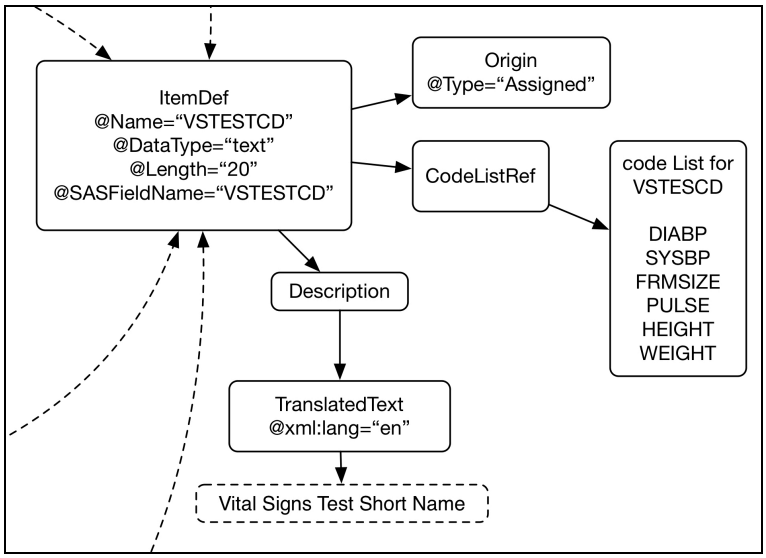

Figure 6: VSTESTCD in Define.xml

*Note: As per **Figure 5** the code list XML has not been expanded.*

So we can see that this one block of XML is defining the variable VSORRES that has generic properties but takes on a particular form when VSTESTCD takes a value of HEIGHT. Figure 3 also includes a second variant for VSORRES and two variants for VSORRESU.

We can step back from these very detailed pictures. ***Figure 7*** takes ***Figure 3*** and overlays the essential ideas or concepts we are dealing with. We have three variables VSORRES, VSORRESU and VSTESTCD with two variants for each of VSORRES and VSORRESU that are valid depending on the value of VSTESTCD.

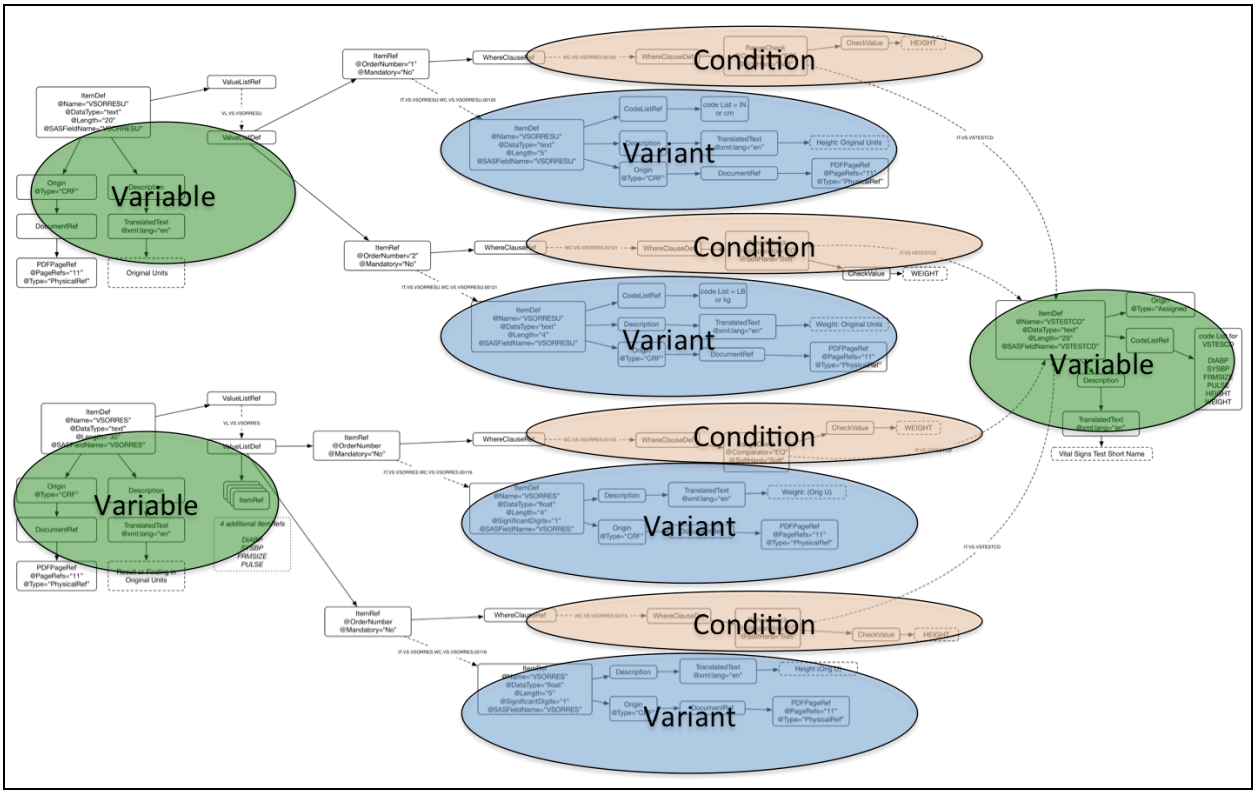

Figure 7: Essential Concepts

Figure 8 takes this high-level view from ***Figure 7*** and represents the same information as a graph. Additional information has been added to align with the logical model presented in ***Figure 2*** with the addition of the Domain node to link the variables VSTESTCD, VSORRES and VSORRESU together as they would be within a fully populated model. The four variants are present with the conditions shared between the variants, as they have common properties. These variants then link back to the variable VSTESTCD that they target. From this figure, the non-rectangular nature of the information and the utility of the graph starts to emerge.

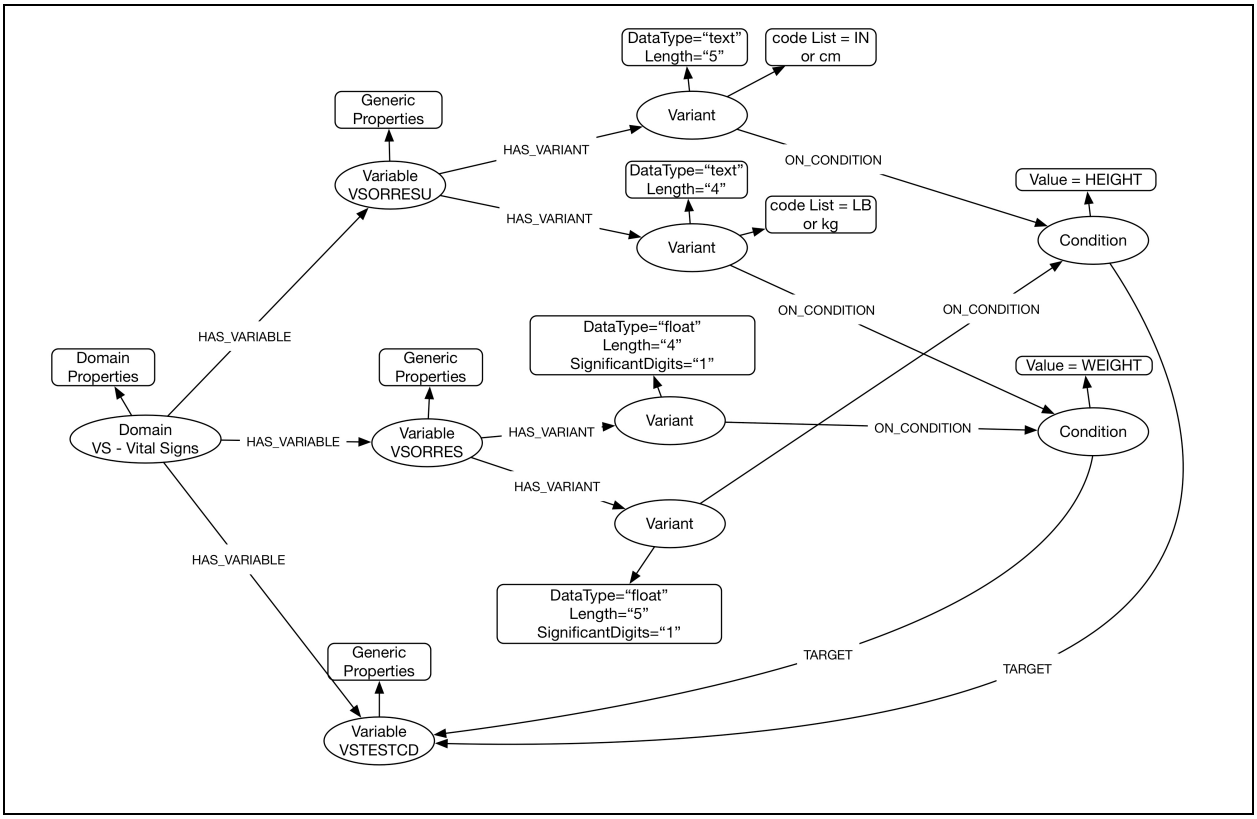

Figure 8: VLM Graph for Vital Signs

To emphasize the non-rectangular nature, the next ***Figure 9*** illustrates the populated graph for the small Laboratory examples given above in the earlier bullet list. This figure does not include the units and LBORRESU but we can already see the increase in the number of relationships that exist when describing the VLM for three tests. Note that, once again, there is re-use of the conditions where appropriate.

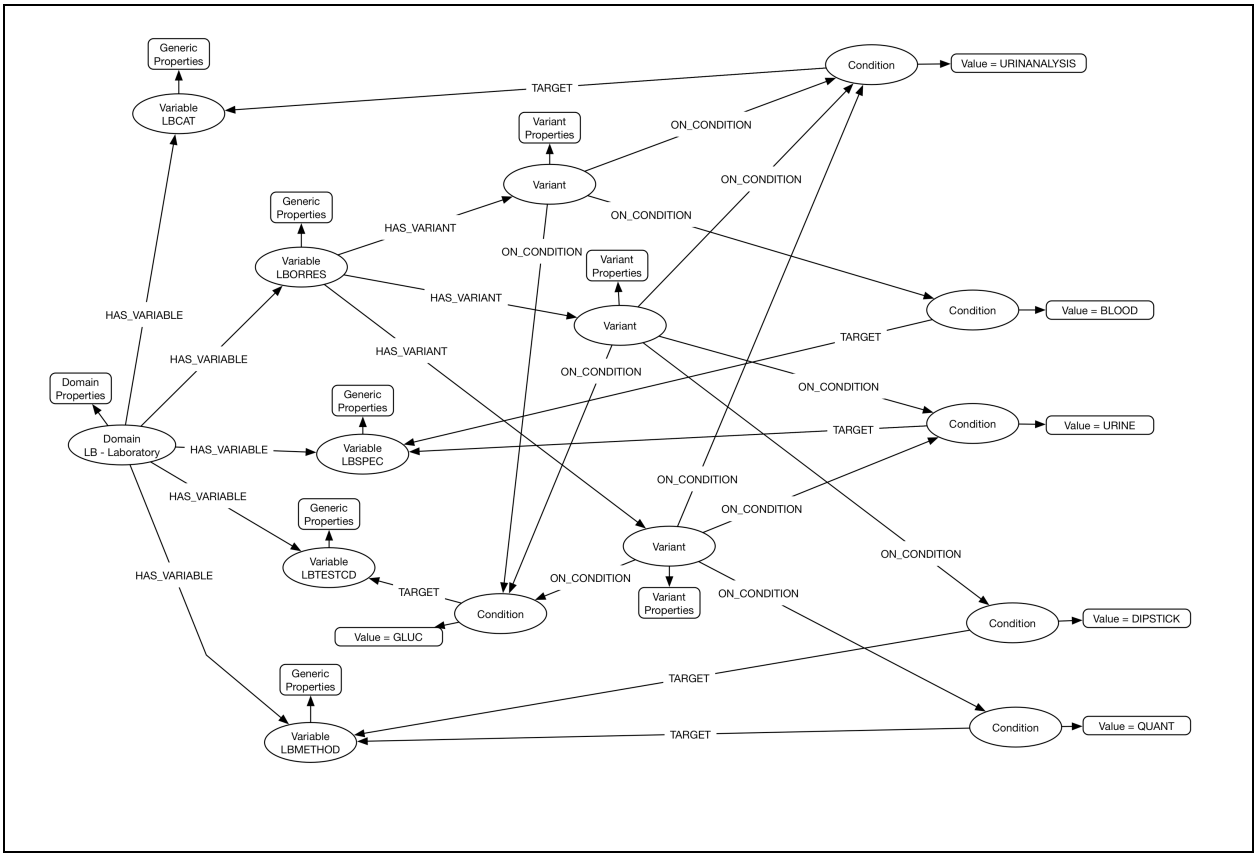

Figure 9: VLM Graph for Laboratory

One significant advantage of the graph is the ability to “walk it”. So in the Vital Signs example in ***Figure 8*** we could start with VSORRESU and detect there are two variants and the conditions for those variants, i.e. rebuild our opening statements of:

- VSORRESU
  - datatype=”text”, length=”5”, code list=”IN” or “cm” when VSTESTCD = HEIGHT
  - datatype=”text”, length=”4, code list=”LB” or “kg” when VSTESTCD = WEIGHT

However, we can also start at VSTESTCD and traverse the graph in the other direction and build a statement such as:

- When VSTESTCD = HEIGHT
  - VSORRES has datatype=”float”, length=”5”, significant digits=”1”
  - VSORRESU has datatype=”text”, length=”5”, code list=”IN” or “cm”

This second form is important, as it is a similar construct to Biomedical Concepts (BCs) and as such allows VLM to be populated from a Metadata Repository (MDR) holding such definitions. BCs have been discussed in previous work by the author (Iberson-Hurst, 2015) though at the time BCs were referred to as Research Concepts. In a simple diagrammatic form, a BC from the MDR would look something like that shown in ***Figure 10***. We can see the same information is present in all of the representations we see here, it is the presentation of the information that is changing.

As an aside, you will notice the difference in the representation of the submission value for inches used in the examples. Above we have used “IN” while in ***Figure 10*** we have “in”. This was checked in the MDR and it was noted that the submission value has been changed by CDISC as indicated in ***Figure 10***. A small and simple example but an illustration that, by having access to the MDR, we have the information available to make an informed decision about which value to use given which version of the terminology we wish to use.

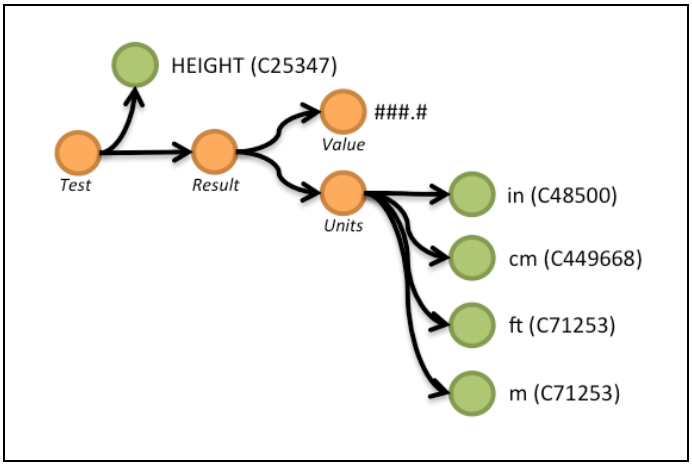

Figure 10: Biomedical Concept For Height

Date	Identifier	Submission Value
2014-03-28	C48500	IN
2014-06-27	↓	↓
2014-09-24	↓	↓
2014-10-06	↓	↓
2014-12-16	↓	IN in
2015-03-27	↓	↓

Figure 11: Changes to C48500 Submission Value

Having the ability to see the VLM in multiple ways allows the presentation of such information to be tailored to the user and make it easier for the user to select the right metadata for inclusion in the study definition.

Another aspect that may have been noted is that the VLM from the MDR holds more values than are used within the study. Nothing wrong with that; the BC is the super-set and it is expected that a study would constrain such terminology subsets, as required by study needs. But having the information in the MDR again allows it to be available to users to make informed decisions and automated checks to be run as necessary within tools.

THE CORRECT GRAPH?

It is worth stepping back at this point and ask a hard question. Is this, or any define.xml graph, the right graph? Should we be modeling define.xml? The answer short-term is yes, we have issues with define.xml today, these need to be resolved as this is the reality of our work today. Longer term, the answer is no. With better metadata earlier in the life-cycle it is apparent that define.xml could be dispensed with altogether. The work presented in this paper is about solving today's issue rather than tomorrow's.

TOOLING

GENERAL

The define.xml tool is divided into a number of functions:

- An ability to load a study definition with content from:
  - a Study Build; or
  - an existing define.xml file.
- Build a study definition from scratch.
- A set of source definitions such as Domains, Terminology, SDTM versions and BCs.
- Manage and maintain study definitions and versions thereof.
- An ability to serialize a study definition to a given define.xml version.

The tool forms part of a set of cooperating tools as shown in the simple architectural diagram in ***Figure 12***. The define tool uses an Application Programming Interface (API) to access a MDR such that the tool itself does not need to maintain all of the CDISC and sponsor metadata such as domains and variables definitions, the CDISC Terminology and VLM. The tool can access electronic definitions of study builds via a second API to a Study Build Tool.

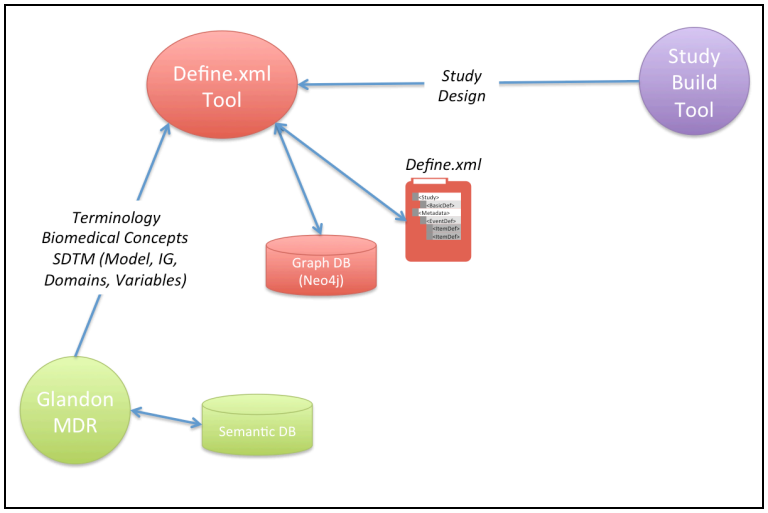

Figure 12: Tool Architecture

LOAD STUDY DEFINITION

The tool is designed to load a study definition from a study build file or an existing deifne.xml file. In either case, the logical model will be populated on the basis of the content extracted.

For the study build file, the tool examines the study definition, uses metadata to determine where collected data would reside within the SDTM structure and selects the domains, variables, VLM and other definitions based on the information found. VLM is sourced either from the study definition via the Biomedical Concepts used or, where concepts are not used, other techniques are used to try and reduce the workload on the user.

For the define.xml file case the content of the file will be reflected in the logical model. When loading from a file obviously there is little or no interpretation of content required.

BUILD STUDY DEFINITON

When neither a study build nor a define.xml file is available the tool needs to maximize the help available to users. In this scenario, the tool will use a number of criteria to help the user in selecting domains and variables that should be within the study definition. Access to VLM than allows further informed choices to build additional content including methods and comments. The whole aim is to utilize the access to high-quality metadata to aid and guide the user.

SOURCE DEFINITIONS

For all of the use cases, it is necessary to have access to the structure and content within the CDISC SDTM Models and Implementation Guides, VLM as well as the CDISC Terminologies. As already stated earlier, such needs can be met by various mechanisms but it is desirable, so as to have access to updates and the latest versions, to have these stored within some form of MDR. Also access to the MDR allows for access to sponsor-defined custom domains and any supplemental qualifiers defined as part of CDISC or sponsor domains.

It is interesting when we look at Comments and Methods. Historically we have seen these entered into MS Excel or other tabular structures and then reused in a cut and paste manner. But methods in particular are but more metadata associated with domains and variables. While we have not touched on this topic, work is being performed to build metadata that can be reused across studies to ease the burden on the user, ease the production of define files and increase the quality of the defines files so produced.

STUDY DEFINITIONS

Once a study definition has been created the tool allows for the version management of the definition created. As such new versions can be created and amended and allow for change history to be maintained.

SERIALIZATION OF STUDY DEFINITION

Having a logical model requires that the model can be transformed into the desired XML. The advantage is that the various define.xml versions can be supported from the same logical content making transformations easier. Also, if necessary, a different output format from XML could be supported, for example JSON or a semantic web format such as turtle.

OTHER ASPECTS

It is worth noting some other aspects of tooling that have not been covered within this paper, as they are not the primary focus. Aspects include:

- support for multiple stylesheets such that resulting define files can be visualized;
- validation of define files generated;
- support for aCRF PDFs and the associated page numbers; and
- support for origin and non-CRF based data.

CONCLUSION

It has been seen how graph technology can aid the user in representing a study in a more logical manner than that we see within the XML of a define.xml and presenting that XML model to a user for the user to populate. The user deserves all the assistance we can possibly give. The graph combined with high-quality definitions for SDTM, Terminology and VLM that can be sourced from a MDR can provide a powerful foundation to allow the user to work at a more logical level dealing with items they are familiar with.

REFERENCES

CDISC. 2005-2017. Define.xml Standard. [accessed 2017 August 28]
https://www.cdisc.org/standards/transport/define-xml

Allard C. 2017. PhUSE. Introduction to the PhUSE CSS 2017. [accessed 2017 August 28]
http://www.phusewiki.org/docs/2017_CSS_US/Introduction_CrystalAllard.pdf

CDISC. 2009. XML Schema Validation for Define.xml [accessed 2017 August 28]
https://www.cdisc.org/system/files/members/standard/foundational/define-xml/definereport_v1_0.pdf

Ibserson-Hurst D, 2015. CDISC Standards and the Semantic Web TT09 [accessed 2017 August 30]
http://www.phusewiki.org/docs/Conference%202015%20TT%20Papers/TT09.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Dave Iberson-Hurst
Assero Limited
58 Third Avenue
Teignmouth
TQ14 9DP
United Kingdom

Email: dave.iberson-hurst@assero.co.uk
Web: www.assero.co.uk & www.a3informatics.com